Федеральное государственное автономное образовательное
учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
«Институт космических и информационных технологий»
Кафедра «Информационные системы»

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ С. А. Виденин
подпись инициалы,фамилия
« _____» ___ _____ 2017 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка мобильного приложения «личный кабинет» для клиентов провайдера «Орион-телеком»

Руководитель	_____ подпись, дата	Е.А.Мальцев инициалы, фамилия
Выпускник	_____ подпись, дата	А.А.Трубинов инициалы, фамилия
Консультант	_____ подпись, дата	А.А.Латынцев инициалы, фамилия
Нормоконтролер	_____ подпись, дата	Ю.В.Шмагрис инициалы, фамилия

Красноярск 2017

Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
«Институт космических и информационных технологий»
Кафедра «Информационные системы»

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ С. А. Виденин
подпись инициалы, фамилия

«__»_____2016 г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы

Студенту: Трубинову Александру Александровичу

Группа: ВКИ 12-13Б Направление 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: «Разработка мобильного приложения «личный кабинет» для клиентов провайдера «Орион-телеком»»

Утверждена приказом по университету № 2836/с от 06.03.2017 г.

Руководитель ВКР: Е. А. Мальцев - старший преподаватель кафедры «Системы искусственного интеллекта» ИКИТ СФУ, консультант А.А.Латынцев – к.т.н., доцент кафедры «Системы искусственного интеллекта»

Исходные данные для ВКР: Список требований к разрабатываемому приложению, методические указания научного руководителя.

Перечень разделов ВКР: Введение, специфика разработки приложения, основная часть, заключение, список использованных источников.

Перечень графического или иллюстрированного материала с указанием основных чертежей, плакатов, слайдов: Презентация, выполненная в Microsoft Office PowerPoint 2010.

Руководитель ВКР _____ Е.А. Мальцев

(подпись)

Консультант _____ А.А. Латынцев

(подпись)

Задание принял к исполнению _____ А. А. Трубинов

(подпись)

« ___ » _____ 2016г.

РЕФЕРАТ

Выпускная квалификационная работа по теме: «Разработка мобильного приложения «личный кабинет» для клиентов провайдера «Орион-телеком»» содержит 51 страницу текстового документа, 20 рисунков, 34 использованных источника.

ИНФОРМАЦИОННЫЕ СИСТЕМЫ, МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ, ВЕБ-РАЗРАБОТКА

Целью разработки мобильного приложения «Личный кабинет» является создание удобной площадки доступа к личному кабинету клиентов компании с возможностью получения информационной справки при пользовании услугами компании.

Основные задачи:

- Выявление и анализ требования к мобильному приложению
- Изучение требований компании и работы сотрудников
- Выбор средств разработки мобильного приложения
- Создание графического интерфейса приложения
- Создание серверного приложения
- Интеграция приложения и сервера для работы с имеющимися программными инструментами в компании

Основные результаты:

- Создано мобильное приложение удовлетворяющее требованиям
- Мобильное приложение было протестировано в реальных условиях компании «Орион телеком»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Средства связи прогрессируют с каждым годом, каких то десять лет назад у большинства пользователей были обычные мобильные телефоны, которые хорошо справлялись с задачей быть средством коммуникации людей. При этом уже тогда начали появляться первые смартфоны, которые включали в себя функциональность: камеры, фонарика, плеера интернет браузера и т.д. Сейчас смартфоны очень плотно вошли в нашу жизнь, ими пользуются как дети, так и пенсионеры, но основной пласт потребителей люди в возрасте 18-40 лет. Сейчас смартфон это маленький карманный компьютер, который догоняет по своим характеристикам больших стационарных собратьев, на который можно установить множество различных приложений как обучающих, так и игровых.

В мире на данный момент выпускается очень много разнообразных устройств, таких как: смартфоны, планшеты, видео-приставки и т.д. под управлением операционной системы Android. По данным исследовательской и консалтинговой компании Gartner только на 2016й год в мире было продано 1.5млрд. смартфонов, из которых, по меньшей мере, 40% устройств на Android. По какой же причине операционная система Android получила такое большое распространение?

Одним из факторов успеха Android является то, что различные производители выпускают устройства именно для этой системы. Второй фактор это наличие огромного количества бесплатных приложений выпущенных под эту систему, благодаря доступным средствам разработки, которые в большинстве своём бесплатны в то время, как разработка, на другую популярную операционную систему iOS зачастую требует значительных финансовых вложений. Помимо этого разработчику Android доступны бесплатные библиотеки для работы со сторонними ресурсами, такими как (Google map API, Yandex mapkit и т.д.)

В данной работе будет описано создание гибридного информационного приложения «Личный кабинет» для клиентов провайдера «Орион-телеком» для

мобильных устройств с операционной системой Android, ориентированного на пользователей в возрасте от 18 до 45лет. Перечисленные ранее преимущества определяют многочисленность и большое распространение современных устройств на базе Android. Данный возраст взят т.к. компания «Орион телеком» предоставляет доступ к услугам интернета и кабельного телевидения различным возрастным группам населения и пользователи смартфонов под управлением операционной системы Android также находятся в различных возрастных группах.

Опишем основные особенности приложения для пользователя с точки зрения информационных технологий.

Мобильные устройства плотно вошли в нашу жизнь, некоторые пользователи совсем отказываются от стационарных компьютеров и ноутбуков в пользу смартфонов, планшетных компьютеров, смарт телевизоров и подобных устройств. Конечно же, есть мобильные версии сайтов, которые вполне удобны для восприятия на небольших экранах, но они имеют ряд недостатков в частности на них нельзя отправлять push-уведомления.

Мобильные push-уведомления это оповещения, которые приходят на смартфон от различных приложений это полезный инструмент, который позволяет сэкономить, например на смс-рассылке. Для удобного оповещения наших абонентов в нашем приложении будут использоваться как pull, так и push уведомления.

Также прослеживается тенденция как крупных, так и небольших фирм, если не полного отказа от сайта, то полноценного дополнения к нему мобильным приложением, это дополнительно говорит о том, что потребность в мобильном приложении для пользователей становиться всё более важной.

Компания Орион телеком предоставляет свои услуги в четырех городах Сибирского федерального округа, это Красноярск, Канск, Иркутск и Абакан. Более 120 000 жителей региона являются пользователями услуг ГК «Орион Телеком»

Рынок услуг информационных технологий в Красноярске – рынок с высоким уровнем конкуренции. Рынок Красноярских интернет–провайдеров насыщен. Крупнейшими поставщиками услуг являются «Ростелеком», «Орион телеком», «Дом.ru», «Билайн», «Telecoma», «ТТК–Сибирь» и другие. Все перечисленные провайдеры активно ведут борьбу за целевой рынок внутри города, а некоторые и за ее пределами. Любому из них важно выделяться на фоне других и как минимум не отставать, часть провайдеров из этого списка уже имеет своё приложение.

В рамках данной работы, были проведены опросы среди менеджеров компании для выявления наиболее частых вопросов при обращении абонентов в офис и по телефону. Опросы и наблюдения показали, что в пиковые часы абонентам, иногда приходиться ждать на телефоне несколько минут, для того, что бы просто уточнить баланс или лицевой счет, в то время как менеджеры и специалисты контактного центра проводят диагностику или же рассказывают об условиях подключения.

1 Задача данного приложения

Учитывая проблематику, было принято решение о создании небольшого информационного приложения, пользователям которого не приходится стоять в долгих очередях для того, чтобы получить консультацию. Нужную информацию можно получить не только у менеджеров, которые ответят на все вопросы, но также будет возможность уточнить информацию по балансу, лицевому счету и т.д. в мобильном приложении компании.

1.1 Постановка задачи

Целью данной работы является создание удобной площадки для доступа к личному кабинету клиентов компании «Орион телеком» с мобильных устройств. Для реализации главной цели необходимо разработать мобильное приложение «Личный кабинет» для устройств под управлением операционной системы Android. Основная задача приложения информационная справка абонентов при использовании услуг компании «Орион телеком». Для этого в рамках данной работы должны быть решены следующие задачи:

- Выявить и проанализировать требования к приложению
- Разработать техническое задание
- Разработка графического пользовательского интерфейса.
- Создание сервера для работы приложения
- Интеграция серверного приложения для работы с программными инструментами в компании

1.2 Серверное приложение

- Необходимо реализовать серверное приложение, обеспечивающее работу мобильных приложений системы.
- Выбранный язык для реализации: JavaScript.

- БД абонентов, синхронизация БД с сервером «Abonents» при помощи API

1.3 Требования к документации

В процессе работы над мобильным и серверным приложениями, в рамках данной данной работы необходимо подготовить следующие документы:

- Техническое задание
- Программа и методика испытаний

1.4 Основные функции приложения

- Предоставление информации о балансе
- Предоставление информации о тарифе
- Предоставление информации о лицевом счете
- Предоставление информации о состоянии обещанного платежа
- Уведомление абонентов о скорой приостановке обслуживания
- Обновление данных с сервера в фоновом режиме

Общая схема функций приложения представлена на рисунке 1.

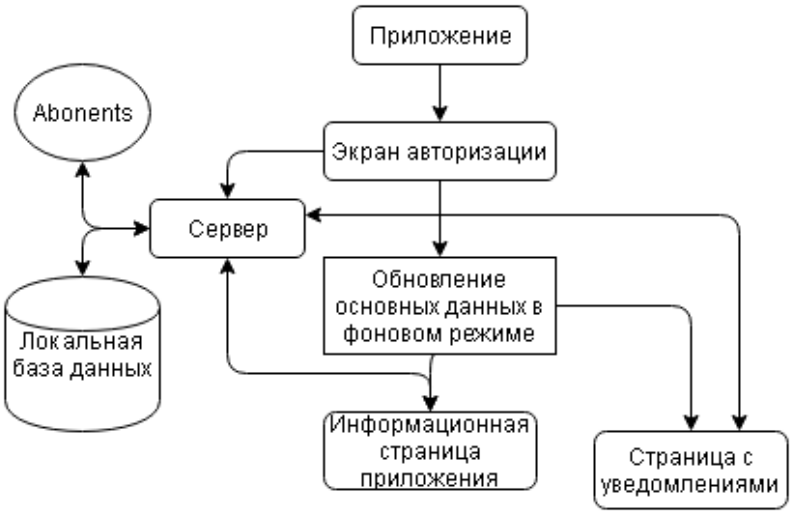

Рисунок 1 - Схема функций приложения

1.5 Особенности разрабатываемого приложения

Учитывая поставленные задачи, приложение имеет специфические особенности, которые опишем ниже.

Приложение является информационным, этот факт ставит задачу о необходимости постоянно отображать корректные данные. Так как приложение содержит несколько полей, которые могут обновиться, необходимо обновление всех полей при входе в приложение и установка таймеров для обновления этих значений:

- Баланс: раз в 5 минут
- Тариф: раз в сутки
- Обещанный платеж: раз в час
- Уведомления: раз в час
- Лицевой счет: никогда

Как отображено на общей схеме функционирования приложения (рисунок 1) обновление данных происходят вне зависимости от самого приложения, это в свою очередь говорит о том, что приложение всегда будет отображать корректные данные.

Разработка велась с использованием контроля версий, что позволило разрабатывать приложение модульно, параллельно разрабатывая как клиентское приложение, так и серверную часть.

1.6 Техническое задание

- Выявить и проанализировать требования

Требования к клиентскому приложению:

- Приложение поддерживает возможность работы в портретной ориентации экрана.
- Локализация приложения предусматривает русскую версию пользовательского интерфейса.

- Версия поддерживаемых Android устройств выше Android SDK 4.4

- Разрешения экранов Android: mdpi (320x480 px), hdpi (480x800px), xhdpi (720x1280px), xhdpi (768x1280px).

- Основная задача приложения информационная справка абонентам при пользовании услугами провайдера «Орион телеком»

- Мобильное приложения для Android должно реализовываться нативными средствами, либо на основе платформы, обеспечивающей компиляцию в нативные приложения для обеспечения максимальной производительности.

- Дизайн мобильного приложения должен быть выполнен в одном стиле с основным сайтом, при этом разработчик должен придерживаться «материального дизайна»

- Для входа в приложение используются логин и пароль от web-версии личного кабинета

- За основу мобильного приложения берется аналог web-версии личного кабинета.

- Открытие левого меню осуществляется либо по нажатию кнопки «Меню», либо swipe с лева на право.

Во время подготовки технического задания были сделаны прототипы приложения на рисунке 2, отображена страница ввода логина и пароля. На рисунке 3 отображена главная страница приложения, то, что пользователь увидит после ввода логина и пароля. Прототип одобрен заказчиком.

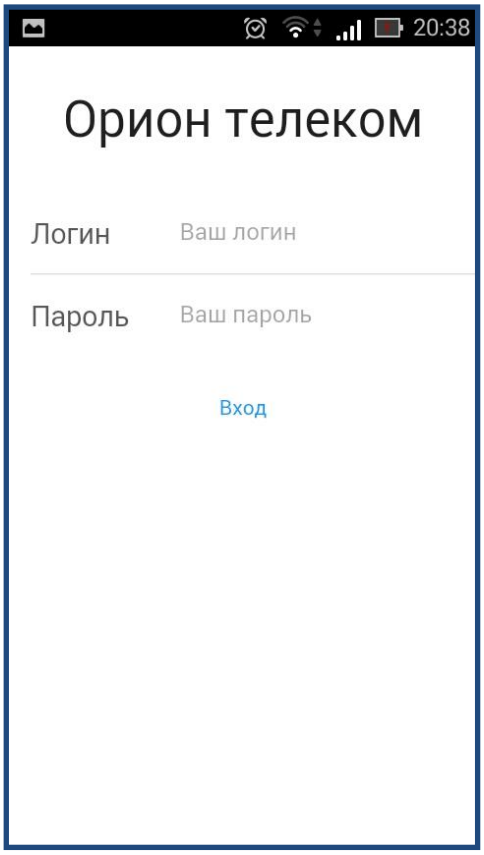

Рисунок 2 – Экран авторизации

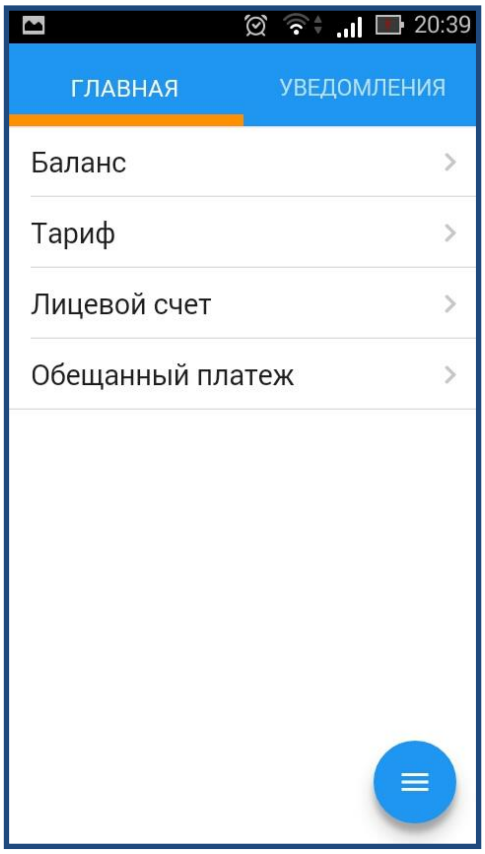

Рисунок 3 – Главное меню

Требования к серверному приложению:

- Необходимо реализовать серверное приложение, обеспечивающее работу мобильного приложения.

- Возможные языки реализации: PHP, Pythoqn, Java, JavaScript.

2 Специфика разработки приложения

2.1 История Android

Как это нередко случается в информационных технологиях, без исторической справки возникновения программы, невозможно объяснить некоторых нюансов, по этой причине сделаем исторический экскурс в развитие операционной системы Android.

Android начали разрабатывать в 2003году, недавно сформировавшейся компанией под названием Android Inc. Google в 2005 году приобрела эту компанию. В это время и была определена специфика архитектуры Android. Вливание финансовых ресурсов и уже имеющихся наработок компании Google во многом повлияли на архитектурные концепции будующей операционной системы. Ниже приведены некоторые примеры.

В начале 2000х годов AJAX приложения пользовались достаточно большой популярностью. Так как Android приложения архитектурно больше соответствует стандартному AJAX приложению нежели к десктомному графическому приложению которое было написано на языках Java, C#, C++ и т.д. можно предположить что основное влияние на архитектуру Android повлияли именно популярные в то время AJAX приложения.

В то же время выходили полнофункциональные онлайн приложения такие как: Google Docs или Gmail и подобные решения были стандартом. Это не является плохим или хорошим решением, но отличия системы Android от десктопных приложений нужно учитывать при разработке.

В 2008году официально вышла первая версия операционной системы Android и вместе с ней первый полноценный пакет разработчика под названием SDK. Наличие пакета SDK в свою очередь очень сильно повлияло на возможность разработчикам создавать приложение под данную операционную систему

За 2009 год было выпущено 4 обновления системы такие как: 1.1, 1.5 «Cupcake» , 1.6 «Dunut».

В октябре 2009года была выпущена втора версия операционной системы Andoid 2.0 которая поддерживала HTML5 спецификацию, это очень важное дополнение которое в будущем позволит нам создавать гибридные мобильные приложение.

До 2011 г также выпускались версии операционной системы, но в 2011г была представленна первая универсальная платформа Android 4.0 которая подходила как для смартфонов так и для планшетов

Android 4.4 была выпущена в 2013году, выпускалась для устройств с минимальными параметрами ОЗУ 512 мбайт. Также Android 4.4 принесла материальный дизайн частично используется в нашем приложении.

В 2014 выпущено обновление Android 5.0 с обновлением материального дизайна.

2015г выпущена версия Android 6.0

В 2016 анонсирована версия Android 7.0

2.2 Конкуренты системы Android

Раньше, рынок мобильных операционных систем делили такие ОС как:

- Symbian
- Blackberry OS
- Android
- iOS
- Bada

Как видно из этого списка было 5-6 основных производителей ОС для мобильных устройств. На данный момент это две лидирующие операционные системы iOS и Android. В отчете за 2016год исследовательская фирма IDC сообщила о том, что доля устройств Android составляет 85% в то время как iOS 14.3%. Прогнозы IDC относительно рынка операционных систем для

мобильных устройств говорят о том, что доля Android устройств будет увеличиваться. Эти прогнозы отображены в таблице 1.

Таблица 1 - Поставки мобильных устройств

Платформа	Поставки 2016(млн/шт)	Процент доли рынка, %	Увеличение объема в 2016, %
Android	1228,8	85	5,20
iOS	206,1	14.3	-11
Windows Phone	6,1	0.4	-79
Остальные	4,5	0.3	-50
Всего	1445,5	100	0.6

2.3. Архитектурные уровни Android

Имеется три основных уровня операционной системы Android, они разительно отличаются друг от друга, перечислим их:

- Фундамент системы это модифицированная версия Linux.

- Выше уровня Linux располагается уровень который содержит в себе Dalvik VM(виртуальную машину), web-браузер, SQLite базу данных и другие инфраструктурные решения API Java, это уровень инфраструктуры.

- Самый верхний уровень это уровень Android - приложений, разработанных в Google это некоторое расширение уровня инфраструктуры, поскольку сторонний разработчик может использовать эти приложения или их части как строительные блоки для собственных разработок.

2.4. Виртуальная машина Dalvik

Dalvik VM отличается от других виртуальных машин разработанных на языке Java следующими параметрами:

- Для хранения двоичных кодов, используется специализированный формат данных DEX

- Во время разработки была проведена оптимизация виртуальной машины Dalvik, оптимизация позволяет одновременно выполнять несколько процессов.

- В виртуальной машине Dalvik используются архитектурные решения, основанные на регистрах, а не на стековой архитектуре как это сделано в других виртуальных машинах разработанных на Java дает уменьшение бинарных файлов и ускоряет работу виртуальной машины

- Для виртуальной машины Dalvik был разработан индивидуальный набор инструкций

- В одном процессе есть возможность запустить несколько независимых приложений

- Выполнение приложения может затрагивать несколько процессов виртуальной машины Dalvik «естественным образом»

- Создан специальный механизм сюрилизации объектов с помощью которого данные занимают меньший объём и более терпимы к версионным изменениям.

- Создан индивидуальный способ, выполнения вызовов между процессами который, основывается на языке описания интерфейсов Android.

В Android используется архитектура, являющаяся фреймворк-ориентированной на замену вольной архитектуре.

Приложения, использующие вольную архитектуру разрабатываемые на языке Java в начале имеют метод *main(),* а разработка практически полностью зависит от пожеланий программиста.

Фреймворк-ориентированные приложения основываются на существующем фреймворке. Их разработка минимизируется расширением определенных классов или изменением и разработкой определенных интерфейсов, которые предоставлены фрейморком. Эти приложения могут запускать вне фрейморка и без него.

При использовании фреймворк-ориентированной архитектуре разработчикам предписывается корректная последовательность действий, при этом вероятность повторения кода сводиться практически к нулю так как приходиться тщательно следовать шаблонам.

Одной из причин разработки собственного фреймовика была потребность поддержки специализированой системы управления памятью Android

Для взаимодействия между интерфейсом пользователя и логикой приложения в Android используется архитектурный шаблон «Model-View-ViewModel» (MVVM)

MVVM – это одна из лучших для создания пользовательского графического интерфейса.

Архитектура MVVM была создана с целью разделения труда дизайнера и программиста, это решается ясным разделением ответственности:

- Разработка графического интерфейса пользователя производиться дизайнером с использованием технологий, «естественных» для таких работ(XML)

- Логика интерфейса пользователя реализуется разработчиком как компонент ViewModel

- Функциональные связи между пользовательским интерфейсом и ViewModel реализуются через привязки, которые, по сути, являются правилами наподобии «если кнопка А была нажата, должен быть вызван метод onButtonAClick() из ViewModel». Привязки могут быть написаны в коде или описаны декларативным путём (Android использует оба типа).

- Архитектура MVVM используется в том или ином виде большинством современных программ и технологий, например Microsoft WPF и Silverlight, Oracle JavaFX, Adobe Flex, AJAX.

2.5. Составные части приложения Android

Обычно приложение Android включает в себя:

- Java-классы, которые, по сути, являются подклассами основных классов из Android SDK и Java-классов, у которых нет «родителей» в Android SDK.

- Манифест приложения

- Ресурсов вроде строк, изображений и т.п.

- Файлов

На диаграмме (рисунок 4) отображена иерархия базовых классов из Android SDK.

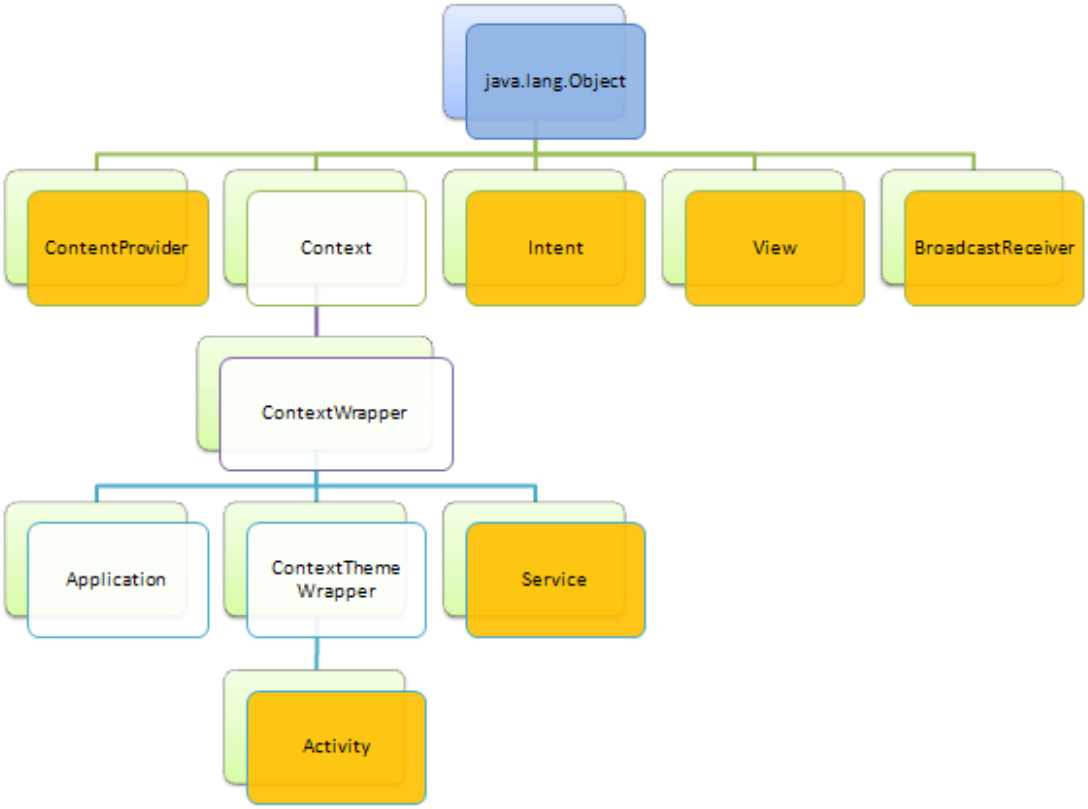

Рисунок 4 - Иерархия базовых классов из Android SDK

Классов в Android, конечно же, больше чем показано на диаграмме, но основные отображены на диаграмме выше (Рисунок 4). Желтым цветом выделены те классы, с которыми разработчику необходимо работать, как и с наследуемыми от них. Остальные классы используются разработчиками напрямую гораздо реже.

View это основной класс для всех виджетов пользовательского интерфейса. Экземпляры наследники класса view составляют собой дерево из которого составляется интерфейс приложения созданного для операционной системы под управлением Android. Это дерево можно создать сторонними средствами, но это будет неправильным решением.

Логика которую содержат класс и подклассы Activity, лежит за интерфейсом пользователя. Этот класс является частью ViewModel в архитектурном шаблоне MVVM. Взаимосвязь между Activity и пользовательским интерфейсом является взаимосвязью один к одному. Как правило каждый слой пользовательского интерфейса имеет только один связанный с ним подкласс Activity.

Во время работы Activity может находиться в одном из следующих состояний:

- Активен и выполняется – этот интерфейс пользователя находиться на переднем плане

- Стоит на паузе – данный интерфейс виден пользователю, но фокус находиться не на нём, в данном случае код не выполняется

- Остановлен – данный интерфейс невидим

Model в модели MVVM представлена в Android-приложении классом Content Provider и его подклассами, зачастую это своеобразная обертка над БД SQLite.

Манифест Android – достаточно важная часть приложения для операционной системы Android.

- Манифест задает имя пакета Java для приложения. Это имя пакета служит уникальным идентификатором приложения.

- Манифест описывает компоненты приложения — операции, службы, приемники широковещательных сообщений и поставщиков контента, из которых состоит приложение. Он содержит имена классов, которые реализуют каждый компонент, и публикует их возможности. На основании этих

деклараций система Android может определить, из каких компонентов состоит приложение и при каких условиях их можно запускать.

- Манифест определяет, в каких процессах будут размещаться компоненты приложения.

- Манифест объявляет, какие разрешения должны быть выданы приложению, чтобы оно могло получить доступ к защищенным частям API-интерфейса и взаимодействовать с другими приложениями.

- Манифест также объявляет разрешения, требуемые для взаимодействия с компонентами данного приложения.

- Манифест содержит список классов Instrumentation, которые при выполнении приложения предоставляют сведения о профиле и прочую информацию. Эти объявления присутствуют в файле манифеста только во время разработки и отладки приложения и удаляются перед его публикацией.

- Манифест объявляет минимальный уровень API-интерфейса Android, который требуется приложению.

- Манифест содержит список библиотек, с которыми должно быть связано приложение

Значения ресурсов

Современные приложения с графическим интерфейсом в той или иной степени всегда используют ресурсы, они могут быть следующих видов:

- Картинки

- Слои графического интерфейса (XML файлы)

- Объявления меню (XML файлы)

- Текстовые строки

- Значения файлов:

Приложения, разработанные для операционной системы Android в общем случае, могут использовать следующие типы файлов:

- Файлы «общего назначения»

- Файлы баз данных

- Файлы Opaque Binary Blob (они представляют собой зашифрованную файловую систему, которая может быть монтирована для приложения)

- Закешированные файлы

2.6 Средства разработки

Так как для разработки приложения были поставлены достаточно сжатые сроки, было принято решение о разработке приложения при использовании следующего стека программных решений:

- HTML5
- CSS3
- JavaScipt
- Framework7
- Phonegap build
- Система контроля версий GitHub
- Для серверного решения также был выбран инструмент разработки на языке JavaScript –node.js это отличное решение для большого количества асинхронных запросов.

Рассмотрим подробнее эти инструменты:

2.6.1 HTML5

HTML(HyperText Markup Language) – язык для структурирования и представления содержимого в сети интернет. Пятая версия спецификации была рекомендована к использованию с 2014года, хотя готов стандарт был до этого. Спецификация разрабатывается и поддерживается организациями W3C и WHATWG.

На данный момент HTML это основной инструмент для написания веб-страниц в сети интернет. Для верстки интернет страниц используются специальные теги, но весь спектр HTML-тегов не сможет удовлетворить

потребности веб-дизайнеров в наполнении содержимым интернет. Для этого используются каскадные таблицы стилей css.

HTML разметка - это кирпичики, из которых стоится веб страница. Если не наполнить сайт html-тегами, использовать CSS или JavaScript не будет иметь смысла. На самом деле, чтобы веб-страница была написана корректно, она должна содержать всего пять обязательных компонентов:

- объявление типа документа (DTD);
- корневой элемент HTML - <html></html>
- заголовок документа - <head></head>
- заглавие документа находящегося в заголовке - <title></title>
- тело документа - <body></body>

Говоря другими словами ниже представлен минимальный код который должен присутствовать в документе HTML5

```
<DOCTYPE html>
<html>
    <head>
        <title>Пустой документ</title>
    </head>
    <body>
    </body>
</html>
```

HTML 5 – эволюция стандарта HTML, добавляющая новые теги и ряд новых возможностей браузерам.

Приведу некоторые примеры:

- Чтение/запись файлов на диск (не любые).
- Встроенная в браузер база данных, которая позволяет хранить данные = на компьютере пользователя.
- Многозадачность с одновременным использованием нескольких ядер процессора.
- Проигрывание видео/аудио, без Flash.

- 2D и 3D-рисование с аппаратной поддержкой, как в современных играх.

2.6.2 CSS3

CSS (Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки html.

Определение стиля в CSS, описывающего внешний вид какого-либо элемента или фрагмента веб-страницы, это просто правило, которое сообщает браузеру, что и каким образом отображать: изменить цвет текста на зеленый, изменить размер заголовка на 5пунктов больше, выделить блок синей рамкой, задать фон блоку, отрисовать меню шириной 200 пикселов для списка гиперссылок. Если перевести текст стилей на естественный язык получим следующее: «Браузер, сделай, чтобы вот это выглядело так-то». По факту, определение стиля состоит из двух основных элементов:

- селектор - это элемент веб-страницы, который необходимо изменить браузеру.

- блок объявления - форматирующие команды.

Селекторами могут быть такие части сайта как блок, картинка, часть текста и т.д. В блоке объявления можно задать такие значения как: шрифт должен быть arial, цвет текста должен быть красным, рамка должна быть 2пикселя, поместить картинку в центре страницы и т.д. возможности форматирования практически бесконечны.

2.6.3 JavaScript

Изначально JavaScript создавался для того, чтобы оживить интернет страницы. Программы на этом языке называются «скрипт». В браузере эти скрипты обращаются напрямую к HTML как только загружается страница, они выполняются. Сейчас JavaScript самостоятельный язык программирования со

своей спецификацией которая называется ECMAScript на данный момент актуальна пятая версия спецификация.

JavaScript это интерпретируемый язык программирования. JavaScript может выполнятся не только в браузере, а где угодно, нужна лишь программа - интерпретатор.

2.6.4 Framework7

Так как изначально было задумано создать html5 приложение, был проанализирован ряд решений. Самым очевидным казалось решение использовать библиотеку jQueryMobile, но при создании с помощью этой библиотеки тестовых приложений их отзывчивость была на очень низком уровне, имелась разительная разница, заметная без тестов с нативными приложениями, отклик от кнопок более 300мс и т.д.

Также важным моментом являлось наличие в фреймворке готовых компонентов, таких как: кнопки, боковое меню и т.д. время на повторную разработку которых хотелось сэкономить. К тому же необходимо была производительность, что бы приложение могло конкурировать с нативными приложениями.

Для решения этих проблем имеются готовые инструменты, которые позволяет разрабатывать: быстрые, отзывчивые и функциональные html5-приложения. Эти инструменты называются Framework7 и ionicframework, выбор пал на Framework7 так как в нём на момент разработки имелась лучшая поддержка функциональности приложения для устройств как на Android так и на iOS, что в будущем поможет без труда перенести приложение на другую платформу.

Framework7 который в себе содержит следующие функции:

- High-performance Animation - Анимации получаются действительно очень быстрые

- fast-clicks - решает проблему задержки 300мс без сторонних библиотек

- Template7 - «движок» шаблонов, подобный Handlebars, но занимает меньше памяти работает значительно быстрее

- Dom7 - более быстрая и прозрачная замена jQuery

- Navigation / Router — множество вариантов переходов и их контроля

- Поддержка стилей, включая поддержку material design

- Большой список встроенных компонентов, таких как forms, buttons, list view, pull to refresh, infinite scroll, slide menu итд

- Свайпы - встроенные свайпы на все случаи жизни: вызвать меню свайпом, возврат к предыдущему экрану, удалить элемент и т.д.

Структура приложения следующая:

- index.html - дизайн приложения

- about.html - дизайн дополнительных страниц

- app.js - файл инициализации и настроек приложения

- css/app.css - таблицы стилей

- lib/framework7.js - сам фреймворк

- lib/framework7.material.css - дизайн всех элементов и компонентов в стиле material

При разработке клиентского приложения я получил следующую структуру, которая отображена на рисунке 5:

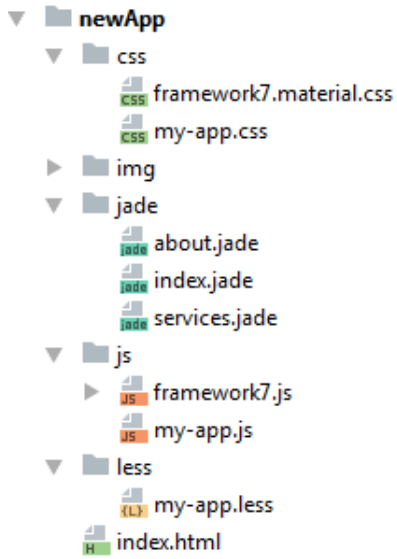

Рисунок 5 – Структура клиентского приложения при разработке

В качестве примера приведу часть кода необходимого, для реализации верхнего меню:

```
<div class="toolbar tabbar tabbar-labels">
  <div class="toolbar-inner">
    <a href="#view-1" class="tab-link active"> // ссылка на первую view
     <span class="tabbar-label">Главная</span></a>
    <a href="#view-2" class="tab-link"> // ссылка на первую view
     <span class="tabbar-label">Уведомления</span></a>
  </div>
 </div>
</div>
```

И пример инициализации приложения:

```
// Инициализируем фреймовик
var myApp = new Framework7({
   swipePanel: 'left' // подключаем левое меню фреймовика
});
// Export selectors engine
var $$ = Dom7; //подключаем библиотеки
// Add views //добавляем вьюв
var view1 = myApp.addView('#view-1');
var view2 = myApp.addView('#view-2', {
   dynamicNavbar: true // включаем динамический navbar
});
```

Вообще Framework7 это кроссплатформенное решение для смены платформы, например на iOS нам потребуется сменить пользовательский интерфейс для этого необходимо будет сменить material.css на ios.css ну и конечно же подправить my-app.css. При какой то нестандартной верстке возможно потребуется немного подправить index.html, но в целом структура приложения остается той же, т.е. при необходимости портировать приложение

например на устройства под управлением iOS нам не потребуется изучать новую технологию, достаточно воспользоваться таким сервисом как Phonegap.

2.6.5 Phonegap build

При выборе платформы для сборки приложений был выбор между тремя подобными друг другу решениями каждое из которых имеет свои особенности и нюансы эти платформы:

- Cordova(CLI)
- PhoneGap(CLI)
- Phonegap build

Зачастую эти решения воспринимают как одно целое, но это не так, для того что бы понять это нужно сделать небольшой исторический экскурс:

В 2009 году была создан PhoneGap внутри молодой компании Nitobi. PhoneGap был создан как среда с открытым кодом, с помощью которой возможно получить доступ к нативным функциям устройства. Цель проекта: обеспечить возможность построения мобильных приложений исключительно на web-технологиях HTML+CSS+JS при этом с возможностью вызова нативного кода.

В 2011 году стартап Nitobi купила компания Adobe первоначальный код был передан компании Apache Foundation. Этот исходный код остался открытым, но ему дали новое имя «Cordova».

Для работы Cordova нужно устанавливать SDK мобильных платформ. Но для облегчения разработки имеется облачный сервис Phonegap build, который компелирует html5 код в приложение, без дополнительных усилий.

Если описывать разницу в двух словах:

Cordova – это фреймовик с открытым кодом, которым управляет Apache.

PhoneGap - это Cordova+инфраструктура от Adobe

- Cordova это OpenSource, а PhoneGap принадлежит Adobe.

- Кроме разных имен пакетов, у PhoneGap и Cordova, разная документация.

- При выходе новых версий ОС (например iOS 64-bit), Cordova обновляется быстрее, чем PhoneGap.

- У Cordova нет облачного сборщика проектов.

При выборе я руководствовался удобством пользования облачным сервисом и интеграцией с системой контроля версий GitHub, по этой причине я использовал PhoneGap build, ниже опишу работу этого сервиса:

Для сборки приложения необходимо

а) Зарегистрироваться на сервере build.phonegap.com,

б) Войти под своей учетной записью

в) нажать кнопку “+new app”

г) ввести адрес репозитория github в котором находиться ваше приложение или же загрузить ваше приложение в архиве .zip(рисунок 6)

Рисунок 6 – Поле ввода адреса репозитория github

д) Выбрать систему, под которую необходимо собрать приложение в нашем случае это Android(рисунок 7)

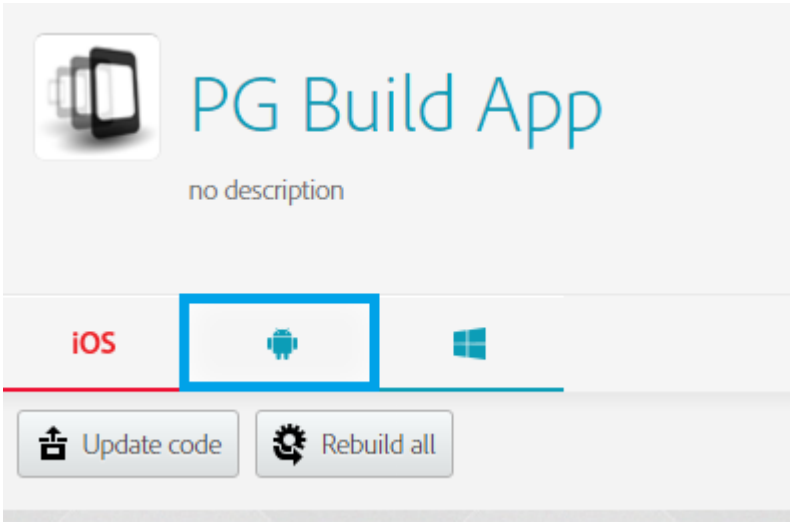

Рисунок 7 – Выбор системы под которую будет осуществляется сборка

Далее сервис скомпелирует программу для операционной системы Android, это файл с расширением .apk, который будет загружен на ваш компьютер.

2.6.6 GitHub

GitHub - крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc

GitHub - это социальная сеть для разработчиков, и по совместительству самая большая площадка для opensource проектов. Позволяет хранить, делиться и разрабатывать проекты многим несвязанным между собой программистам.

Если говорить более простым языком GitHub это облачный сервис позволяющий делать слепок программного кода, сохранять его текущее состояние, изменять сохранять опять и при необходимости восстанавливать старую версию, абсолютно все сделанные слепки храняться на сервере, эти слепки позволяет делать распределенная система контроля версий – Git.

Git — это DVCS, или «распределенная система контроля версий». Фраза «Система контроля версий» говорит о том, что эта система запоминает историю

изменений в файлах которые проходят через неё. Тот факт, что она распределенная означает, что все изменения можно делать и сохранять без выхода в интернет.

Git был разработан Линусом Торвальдсом, тем же человеком который разработал ядро linux, так как в один момент linux переросла систему контроля версий VCS, которая использовалась Линусом изначально для разработки.

Для того что бы объяснить схему работы Git можно представить команду разработчиков, которые каждый день отправляют «патчи» главному разработчику на почту (Патч – это текстовый файл, который перечисляет изменения между двумя версиями каких-то файлов, можно применить новую версию патча к старой версии программы, что бы получилась новая). У главного разработчика также храниться основной код программы. Каждый разработчик может скачать этот основной код и начать работу над его частью, после чего отправить патч на почту главному разработчику, который в свою очередь если всё действительно хорошо работает, добавит патч к основному коду программы. В этом случае Git это просто набор инструментов, для рассылки файлов по этой почте.

Git включает в себя:

- Коммиты
- Деревья
- Ветки

Коммит – это патч, он перечисляет некоторые изменения в некоторых файлах. В нём также есть некоторые заголовки, которые включают в себя автора, отметку о времени и т.д.

Патч отображает различия, между двумя наборами файлов, которые мы будем называть «деревьями». Патч можно применить только к полному набору файлов «дереву» , но после этого мы получим новый набор файлов, новое дерево, по этой причине коммит это также состояние репозитория после применения этого патча.

История Git – это достаточно длинная цепочка инструкций, для пересоздания базы с нуля шаг, за шагом.

В документации Git история отображается слева-направо, по этому история изменений 3 последовательных патчей выглядела бы следующим образом: Патч1---Патч2---Патч3

В Git коммиты обозначаются хэшами, которые имеют примерно такой вид: fb25cd8dcdd513ba55189e81beb297c44e8edc55 но, как правило сокращаются до вида: fb25cd

Хэш также включают в себя SHA-1-хэши патча включая заголовки (и т.к. родитель это один из заголовков, хэш включает в себя хэш родителя, который включает хэш его родителя и т.д. до самого начала)

Отличительное свойство такого хэширования, в том, что отдельно взятый коммит не может быть изменен.

Деревья

Одна из важнейших вещей в Git это дерево директорий, содержащее некоторый ассортимент файлов, любой коммит имеет своё ассоциированное дерево, которое отражает состояние репозитория для этого коммита. Деревья также обозначаются хэшами.

Ветки

Ветки используются для генерации патча, т.е. копируется рабочая директория, в неё производятся изменения и для того, что бы сгенерировать патч, необходимо получить разницу между двумя дирректориями.

Структура работы GitHub показана на рисунке 8.

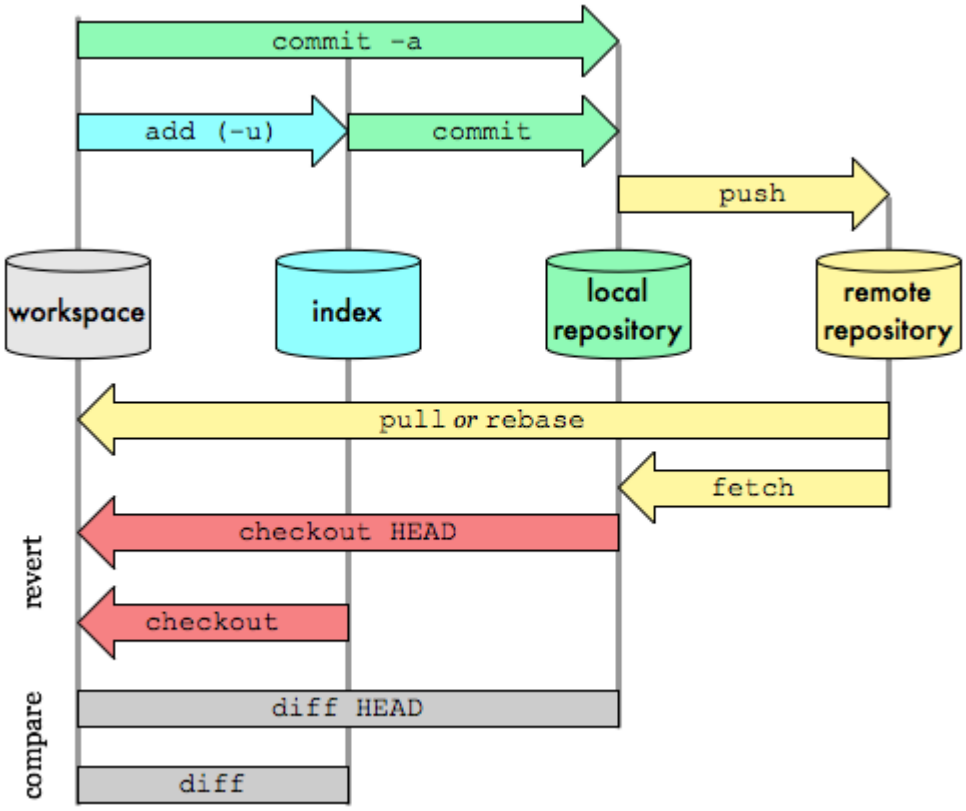

Рисунок 8 – Структура работы GitHub

2.6.7 Необходимое программное обеспечение для работы с GIT

В первую очередь надо установить клиент git: обязательно потребуется консольный клиент, графический клиент можно установить по желанию, исходя из своих предпочтений. На Unix системах можно воспользоваться менеджером пакетов (yum на fedora и подобных или apt-get на debian, ubuntu и подобных) вместо того, чтобы скачивать установщик с сайта. Консольный клиент, аналог командной строки(рисунок 9)

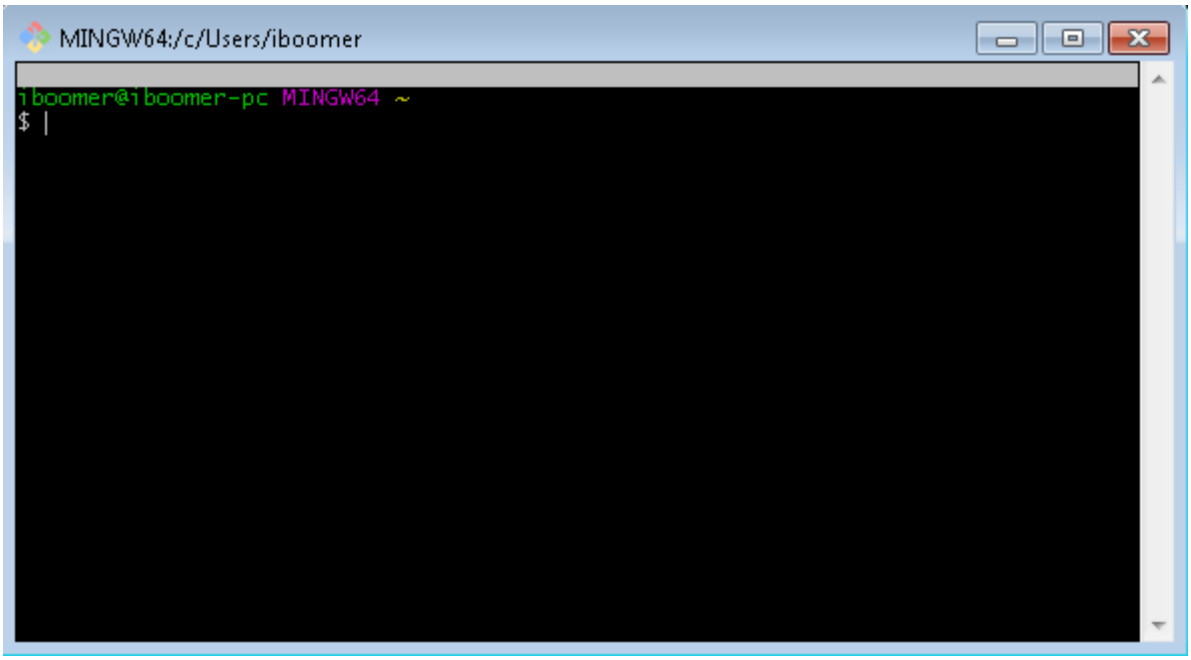

Рисунок 9 – Консольный клиент Git

Если использовать графический клиент, нажатия кнопок сводятся к выполнению определенных команд консольного клиента, но зачастую возможности графических клиентов ограничены по сравнению с консолью.

После необходимо зарегистрироваться на https://github.com/. Далее можно будет создавать свои репозитории или присоединиться к работе над проектами коллег, сделав fork другого репозитория.

Описывать дальнейшие инструкции по работе с Git в данной работе я не буду, так как по этой теме имеется очень много информации, общий принцип работы был описан выше.

2.6.8 Node.js

Node.js это серверная технология, основой которой является движок JavaScript под названием V8, разработанный компанией Google. Это легко масштабируемая система, работающая с асинхронным вводом – выводом а не с отдельными процессами или потоками. Node.js отлично подходит для веб-приложений, если они не выполняют вычислений, но к ним происходят частые

обращения. Обычные веб-сервера, например такие, как Apache, при любом запросе на веб-ресурс выделяют отдельный программный поток или вызывают новый процесс для обработки этого запроса.

Если даже в текущий момент времени Apache достаточно быстро реагирует на подобные запросы, а после завершения запроса всё приводит к изначальному состоянию, при этом сервер задействуется множество ресурсов. Как результат при повышении популярности веб-приложения возникают предпосылки для серьезных проблем производительности. Node же, прослушивает конкретные события и когда эти события происходят реагирует на них соответствующим образом. Node не блокирует никаких запросов, дожидаясь завершения действий, инициируемых событием, а сами события обрабатываются в относительно простом цикле обработки событий по принципу «первым пришел — первым обслужен». Node-приложения создаются с помощью языка JavaScript (или альтернативных языков, компилирующихся в JavaScript), который ничем не отличается от языка, применяемого в приложениях на стороне клиента. Однако в отличие от языка JavaScript, используемого в браузере, для Node нужно создать среду разработки. Node можно установить на платформе Unix/Linux, Mac OS или Windows.

В качестве небольшого примера самый простой сервер на node.js будет иметь следующий код:

```
// загрузка модуля
http var http = require('http');
// создание http-сервера
http.createServer(function (req, res) {
// заголовок контента
res.writeHead(200, {'content-type': 'text/plain'});
// запись сообщения и завершение сигнальной связи
 res.end("Hello, World!\n");
}).listen(8124);
console.log('Server running on 8124');
```

2.7 Среда разработки

2.7.1 Brackets

Brackets позиционируется как текстовый редактор, но по факту он больше похож на полноценную IDE. При стандартной установке мы получаем следующий функционал:

- Плагин для предварительного просмотра, работает только в браузере Google Chrome, меняете код в brackets, сразу же появляются изменения на странице, просмотра, своеобразный wysiwyg редактор.

- Подсветка синтаксиса

- Подсказки при редактировании CSS, HTML, JavaScript файлов

- Плагин для редактирования CSS напрямую из HTML

Внешний вид без дополнительных плагинов отображен на рисунке 10.

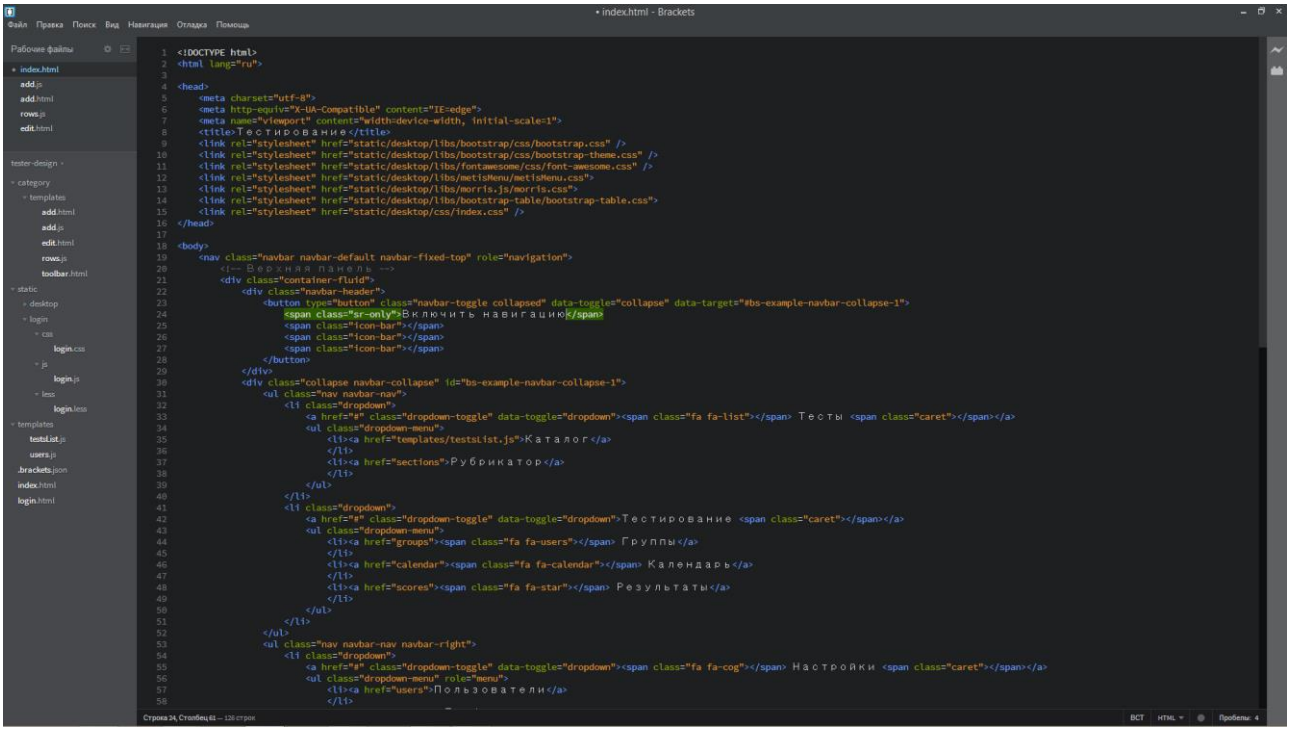

Рисунок 10 – Интерфейс Brackets без дополнительных плагинов

Но огромное количество плагинов, которые были разработаны для Brackets позволяет превратить его в полноценное IDE решение.

2.7.2 WebStorm

WebStorm – среда для разработки на JavaScript, которая подходит как для front-end-разработки, так и для создания приложений на Node.js.

Главное достоинство WebStorm – это удобный и умный редактор JavaScript, HTML и CSS, который также поддерживает языки, такие как TypeScript, CoffeeScript, Dart, Less, Sass и Stylus и фреймворки, например, Angular, React и Meteor.

WebStorm, как и другие IDE, разработанные на основе платформы IntelliJ IDEA, делает разработку проще и удобней, обеспечивая подсветку и авто дополнение кода, его анализ по ходу редактирования, быструю навигацию и переработку кода и предоставляя разработчику мощные инструменты отладки и интеграцию с системами управления версиями. WebStorm «понимает» структуру проекта и код, обнаруживает возможные проблемы и предлагает их решение. Встроенные в IDE инструменты для тестирования и работы с проектом помогут в разработке и сделают ее удобней и продуктивней.

Основные особенности

- отладка кода на JavaScript, а также на ECMAScript 2015+, TypeScript, CoffeeScript и Dart с использованием source maps
- отладка Node.js приложений
- интеграция с системами управления версиями Git, GitHub, Subversion, Perforce, Mercurial, CVS
- интеграция с системами отслеживания ошибок (ESLint, JSHint, JSLint, TSLint, Stylelint)

Интерфейс Webstorm отображен на рисунке 11.

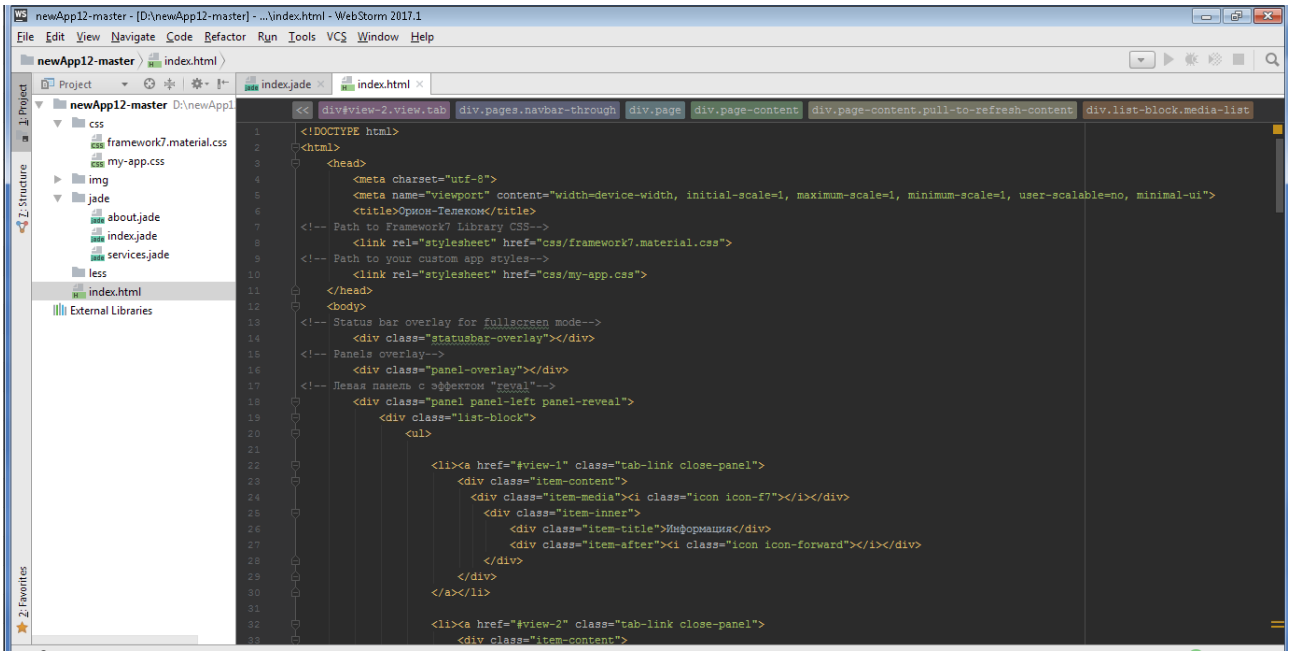

Рисунок 11 – Интерфейс WebStorm

3 Основная часть

3.1 Структура приложения «Личный кабинет»

По итогу выполненной данной работы я создал гибридное мобильное приложение.

Гибридное приложение это мобильное приложение, "запакованное" в нативную оболочку. Устанавливается приложение через магазин приложений Android, имеет доступ к тем же функциям мобильного устройства, что и нативное, но разрабатывается с помощью web-языков HTML5, CSS и JavaScript. Явным отличием гибридного приложения от нативного является легко переносимым между различными платформами, однако немного уступает в производительности.

В общем виде архитектура гибридного приложения отображена на рисунке 12.

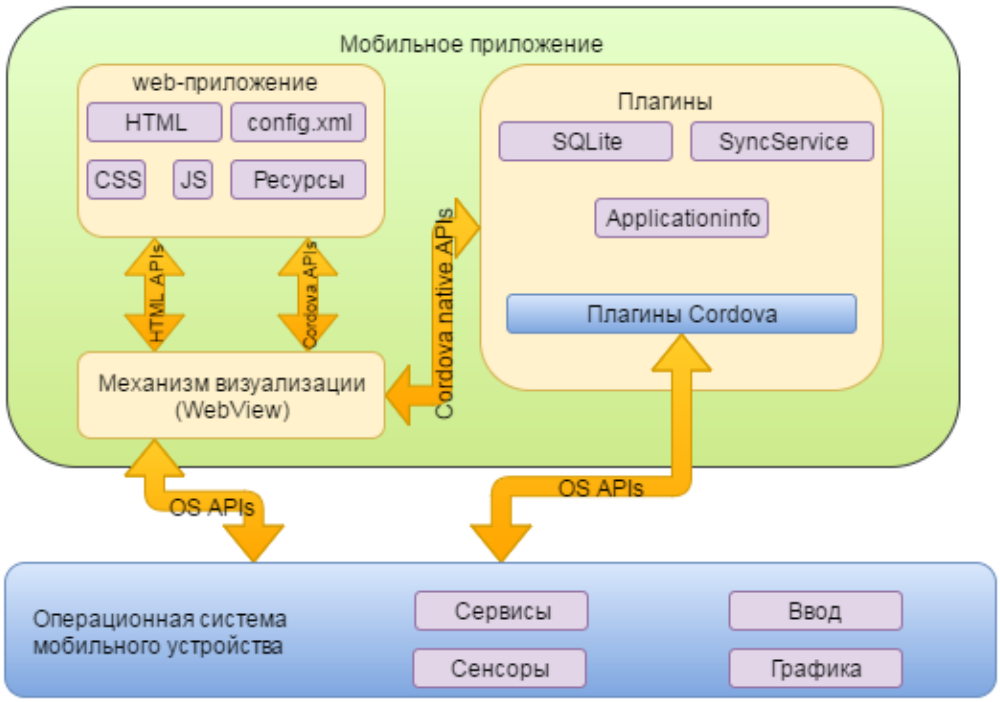

Рисунок 12 – Архитектура гибридного приложения

Мобильное приложение состоит из двух частей это: клиентское приложение, устанавливаемое на мобильное устройство под управлением Android и серверная часть приложения.

При входе в мобильно приложение абоненту необходимо авторизоваться с помощью имеющихся логина и пароля. Логин и пароль задаются при заключении договора с абонентом. Сотрудниками компании Орион телеком эти данные вносятся в БД и хранятся на сервере «Abonents». С помощью сервера Abonents также производятся списания абоненткой платы абонента, устанавливается обещанный платеж и устанавливаются блокировки. Созданный сервер node.js парсит эти данные к себе в базу данных и хранит их обновления происходят, раз в 10 минут.

Страница авторизации абонента, включающая поля логин, пароль и кнопку «вход» отображена на рисунке 13.

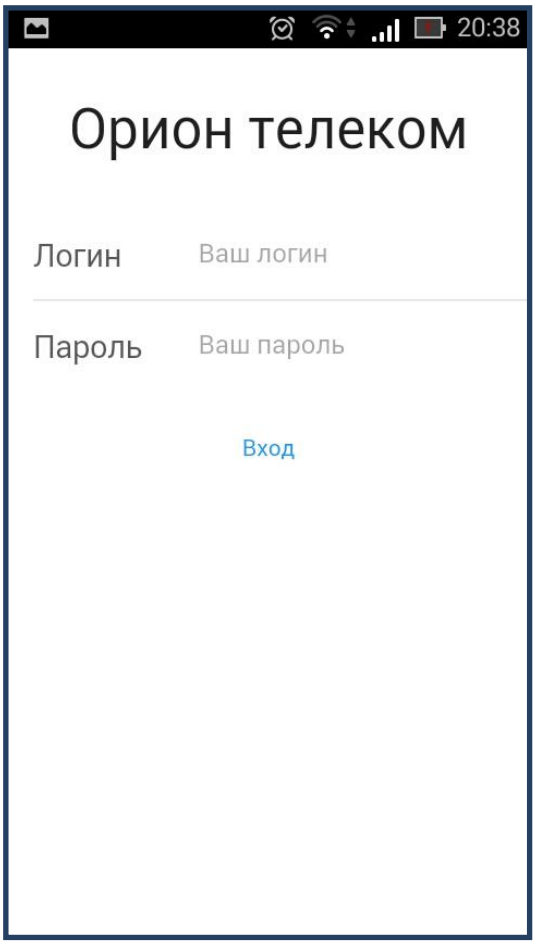

Рисунок 13 - Страница авторизации абонента

Авторизация производиться производится в соответствии с диаграммой, отображенной на рисунке 14.

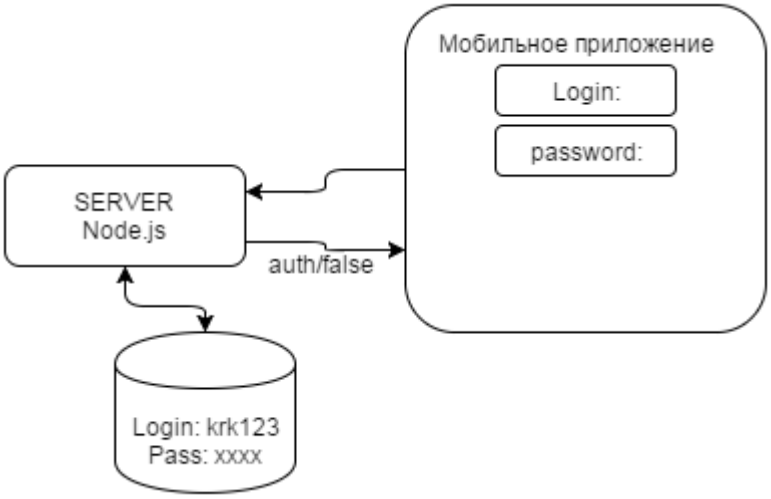

Рисунок 14 – Авторизация абонента

После авторизации клиент увидит страницу, отображенную на рисунке 15.

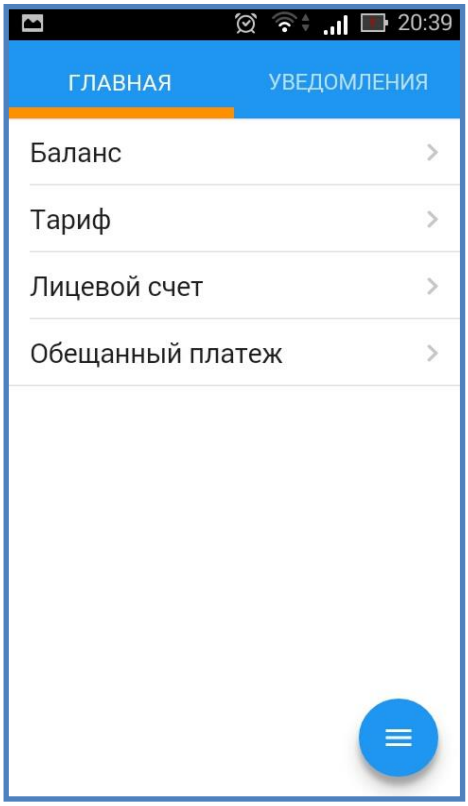

Рисунок 15 – Главная страница приложения

После нажатия на каждое поле будет отображена информация из личного кабинета, это отображено на рисунке 16.

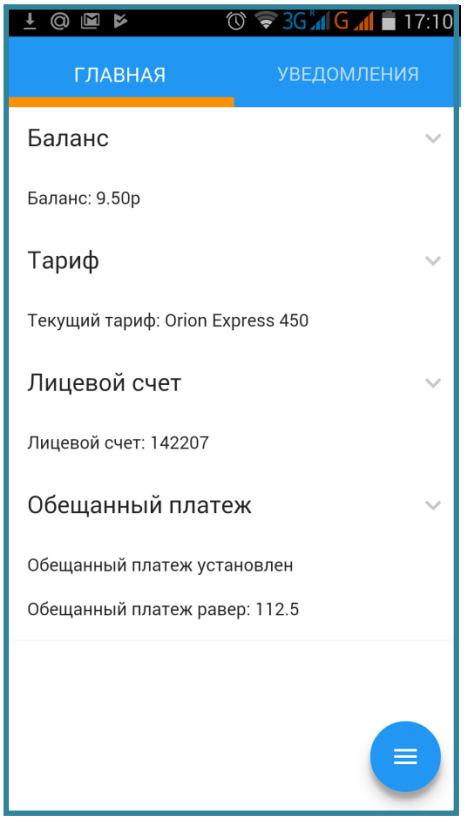

Рисунок 16 – Информация при раскрытии

Данные подгружаются с сервера созданного мной сервера при загрузке приложения и обновления производятся в соответствии с параметрами указанными в техническом задании:

- Баланс: раз в 5 минут
- Тариф: раз в сутки
- Обещанный платеж: раз в час
- Уведомления: раз в час
- Лицевой: никогда

Обновление производится по схеме, отображенной на рисунке 17.

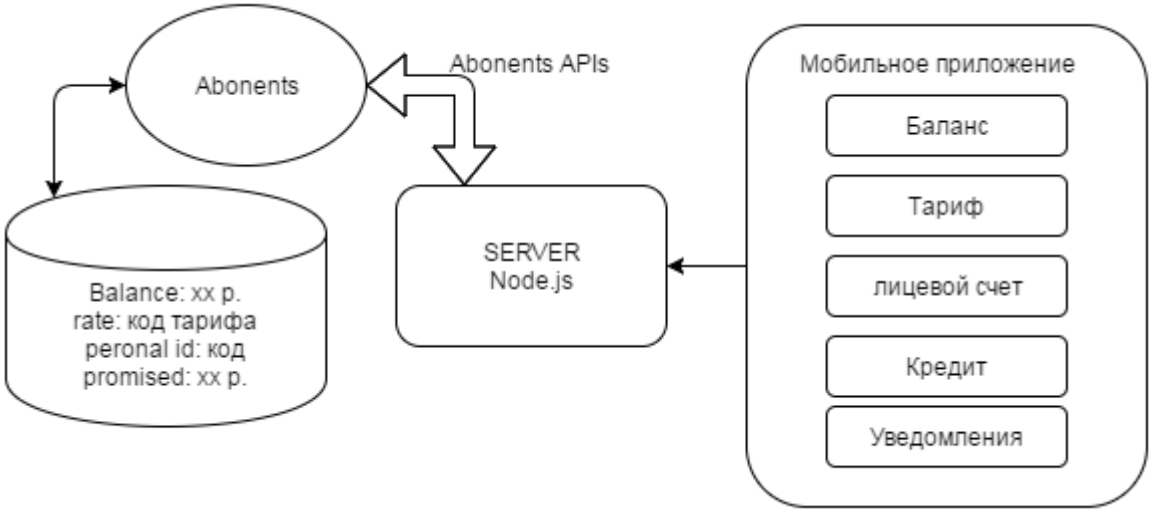

Рисунок 17 – Схема обновления приложения

При нажатии на поле «уведомления» будут показано последнее уведомление для абонента, если потянуть сверху вниз при наличии новых уведомлений они обновятся, рисунок 18.

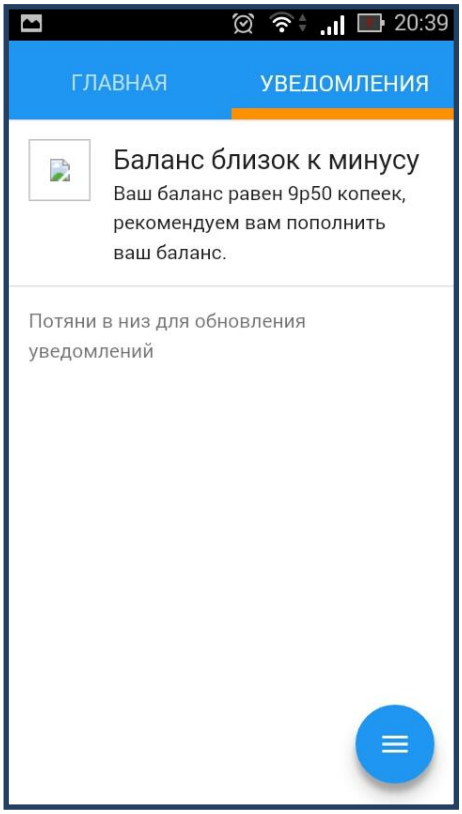

Рисунок 18 – Раздел «Уведомления»

При нажатии на кнопку меню круг с тремя полосками или при свайпе слева направо открывается меню приложения отображено на рисунке 19.

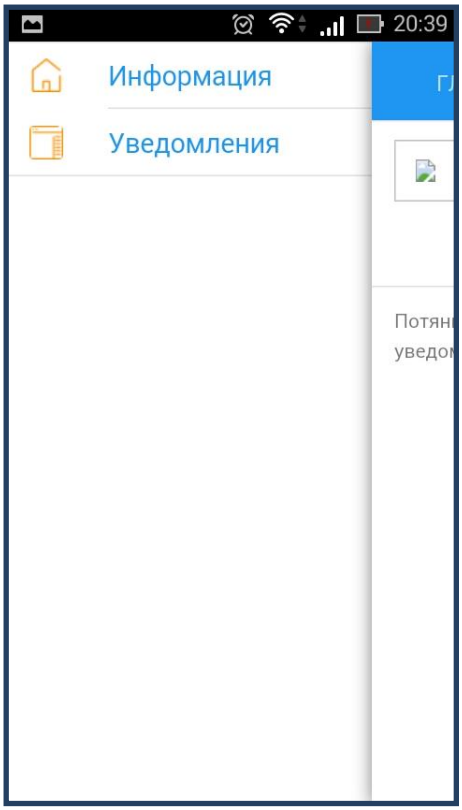

Рисунок 19 – Меню приложения

3.2 Тестирование приложения

В процессе разработки приложения производилось поэтапное тестирование с целью выявления программных ошибок и несоответствий техническому заданию. Для тестирования были задействованы несколько устройств с разными версиями Android, также были созданы эмуляторы смартфона и планшета с разными диагоналями экрана для разных версий Android. Тестируемое приложение последовательно запускалось на этих эмуляторах и устройствах предоставленных для тестирования, производился анализ работы и при необходимости вносились изменения в программный код. Для тестирования отдельных модулей работы с базой данных в код приложения были внесены специальные функции, позволяющие анализировать базу данных

и, производить логирование ошибок, которое отображалось в системном журнале. Они также известны как unit-тесты. Для примера, при изменениях в базе данных проводилась проверка целостности базы данных после чего при необходимости выводилось сообщение в системный log.

Были проведены приведенные ниже тесты.

1 Каждое действие было подвергнуто юнит-тестированию с целью нахождения ошибок, вызванных несоответствием ожидаемых и полученных параметров. Для этого для каждой активности был создан специальный юнит-класс, посылающий в активность различные верные и неверные параметры. При аномальном поведении активности или ее сбое, мною анализировалось поведение и ошибка исправлялась.

2 В базу данных намеренно вносились недопустимые данные в соответствующие поля, которые могли быть неверно интерпретированы программой. Затем мною анализировалось поведение активности во время обработки недопустимых данных.

3 Приложение было запущено на устройствах, работающих под управлением разных версий Android с целью выявления особенностей работы приложения, запущенного в разных операционных системах.

4 Также программный продукт тестировался на реальных устройствах. По результатам тестирования была добавлена виртуальная кнопка «Меню» для устройств, не имеющих аппаратных кнопок.

Тестирование разработанного приложения проводилось на различных версиях операционной системы, краткий список:

- Android 4.4 (KitKat).
- Android 5.0/5.01 (Lollipop)
- Android 6.0 (Marshmallow)

Тестирование происходило на устройствах Android Higjscreen Thor, Google Nexus 5, Билайн Таб2, а также на эмуляторе встроенном в систему Intel xdk тестирование на эмуляторе отображено на рисунке 20

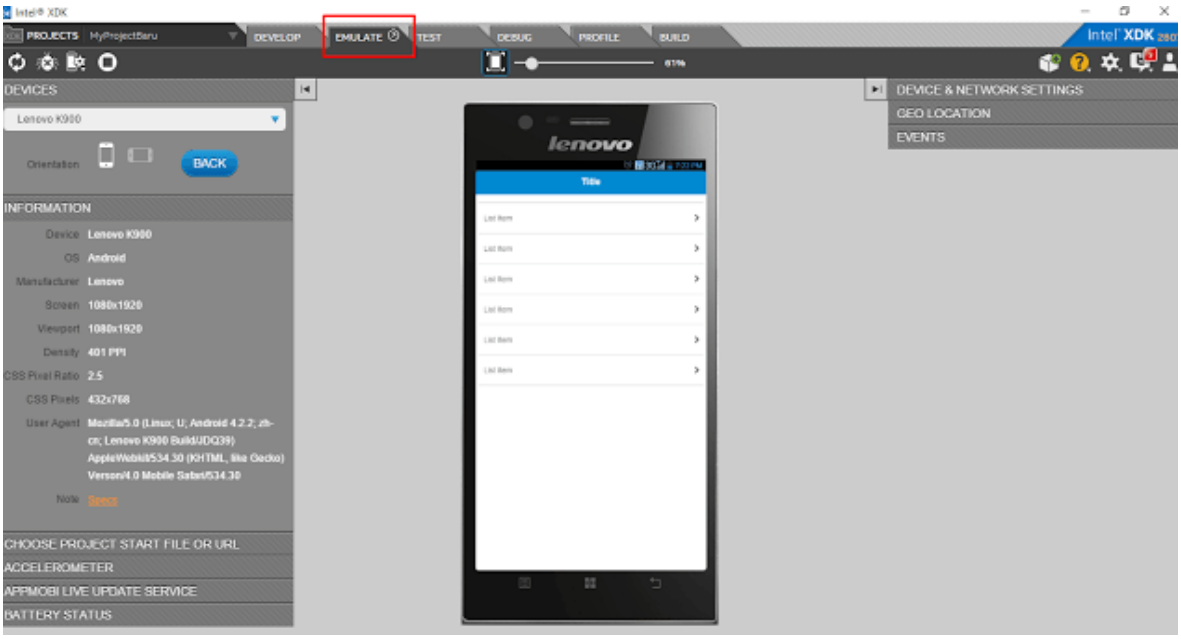

Рисунок 20 – Тестирование на эмуляторе

Тестирование проводилось для различных разрешений экранов:

- 320x280
- 400x240
- 800x420
- 800x480
- 1024x600

В ходе тестирования были выявлены и исправлены проблемы с некорректным отображением информации на некоторых устройствах. В тестировании принимали участия сотрудники компании Орион телеком.

3.3 Развития приложения

Возможными путями развития приложения являются:

1 Интеграция приложения со сторонними сервисами вместо Abonents;

2 Доработка приложения для других популярных операционных систем, таких как iOS и Windows mobile

3 Реализация возможности подгрузки пакетной информации;

4 Добавление функций приложения

5 Добавление онлайн-чата с консультантом

6 Реализация интерфейса на различных языках

3.4 Публикация приложения

Разработанное приложение для тестирования распространялось в виде .apk файла. Запуск и публикация в онлайн – магазине Google play запланирован на конец августа.

ЗАКЛЮЧЕНИЕ

Операционная система без приложений практически не пригодна для пользования, так как в ней отсутствуют такие функции как: просмотр интернет страниц, просмотр уведомлений от различных информационных приложений и подобные функции которые делают из мобильного устройства полноценный мини компьютер. Развитие технологий позволяют разработчикам создавать кроссплатформенные приложения с доступом к нативным функциями системы не прибегая к переписыванию приложений под определенную платформу, этому способствует развитие технологий HTML, CSS, JavaScript и смежных им технологий. Сами приложения могут быть разработаны для удобства пользования той или иной операционной системой, развлечений пользователя, информирования пользования и т.д.

Потребность в разработке мобильных приложений возникает постоянно, так как растет рынок мобильных устройств, появляются новые привлекательные технологии, появляются новые пользователи и появляются новые организации, для которых важно, что бы пользователи их услуг получали информацию в удобном для них виде.

В результате выполнения данной работы был получен опыт в разработке гибридных клиент-серверных приложений, высокую значимость имеет возросший уровень программного кода. Получены навыки работы с популярными и набирающими популярность интернет сервисами, такими как GitHub, Codovra, PhoneGap, Framework7 и другими. Получен глубокий опыт и понимание принципов взаимодействия систем в операционной система Android.

Работая над большими и не очень большими, но важными проектами мы начинаем понимать значимость своей работы и должны показывать хорошие результаты своего труда, так как это несет в себе большую значимость в сфере IT-технологий.

Задание на выпускную квалификационную работу были выполнено в полном объёме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бахирев, А. М. Сюрреализм на JavaScript: учебное пособие / А.М. Бахирев. - Санкт-Петербург: СИНЭЛ, 2014. - 228 с.
2. Берр, Б. jQuery, Подробное руководство по продвинутому JavaScript: 2-е издание.: пер с англ. / Берр Бибо, Иегуда Кац; под ред. А.Галунов – Санкт-Петербург.: Символ-Плюс, 2011. – 624 с.
3. Брокшмидт, К. Введение в разработку приложений для Windows 8 с HTML, CSS и JavaScript 2-е изд.: учебное пособие / К. Брокшмидт – Москва.: Национальный Открытый Университет «ИНТУИТ», 2016. – 459 с.
4. Вейл, Э. HTML5. Разработка приложений для мобильных устройств: пер с англ./ Э. Вейл; под ред. Д.Виницкого. - Санкт-Петербург: Питер, 2015. – 480 с.
5. Гоше, Х.Д. HTML5 для профессионалов.: пер с англ. / Х.Д.Гоше. – Санкт-Петербург.: Питер, 2013. – 496 с.
6. Гудман, Д. JavaScript и DHTML. Сборник рецептов. Для профессионалов.: пер. с англ. / Д. Гудман. - Санкт-Петербург: Питер, 2014 – 523 с.
7. Дари, К. AJAX и PHP: разработка динамических веб-приложений.: пер с англ. / К. Дари, Б. Бринзаре, Ф. Черчез-Тоза, М. Бусика; под ред. А.Галунов. - Санкт- Петербург.: Символ-Плюс, 2007. – 336 с.
8. Дейтел, Х.М. Операционные системы. Основы и принципы 3-е изд.: пер. с англ / Х.М.Дейтел, П.Дж.Дейтел, Д.Р.Чофнес; под ред. С.М.Малявко – Москва.: Бином, 2006. – 340 с.
9. Закас, Н. JavaScript для профессиональных веб-разработчиков.: пер с англ. / Н. Закас. – Санкт-Петербург.: Питер, 2015. – 960 с.
10. Закас, Н. JavaScript. Оптимизация производительности.: пер с англ. / Н.Закас; под ред. А.Галунова. – Санкт-Петербург.: Символ-Плюс, 2012. – 256 с.

11. Кантелон, М. Node.js в действии.: пер с англ. / М. Кантелон, М. Хартер, Т. Головайчук, Н. Райлих; под ред. А.Жданова. - Санкт-Петербург.: Питер, 2014. – 548 с.

12. Лабберс, П. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложения.: пер с англ. / П.Лабберс, Б. Олберс, Ф.Салим; под ред. С.Н. Тригуб. – Москва.: Вильямс, 2011. -272 с.

13. Маккоу, А. Веб-приложения на JavaScript.: пер с англ./ А. Маккоу. - Санкт-Петербург.: Питер, 2012. – 288 с.

14. Макфарланд, Д. Большая книга CSS3. 3-е изд.: пер с англ. / Д. Макфарланд; под ред. Д. Виницкий - Санкт-Петербург.: Питер, 2014 – 608 с.

15. Немцева, Т.И. Компьютерная графика и web-дизайн 1-е изд.: учебное пособие / Т.И. Немцева, Т.В. Казанкова, А.В. Шнякин; под ред. Л.Г. Гагариной. – Москва.: Инфра-М, 2014. – 400 с.

16. О разработке под Android [Электронный ресурс] Блок на хабрхабр посвященный разработке Android - Режим доступа: URL: https://habrahabr.ru/hub/android_dev/

17. Османи, Э. Разработка Backbone.js приложений: учебное пособие / Э.Османи; под ред. А.Кривцов Санкт-Петербург: Питер, 2014. – 352 с.

18. Официальный сайт Framework7 [Электронный ресурс]: Создание интерфейса приложения - Режим доступа: URL: http://framework7.io/

19. Официальный сайт GitHub [Электронный ресурс]: Социальная сеть для разработчиков. - Режим доступа: URL: https://github.com/

20. Официальный сайт для разработчиков Android. [Электронный ресурс] : Режим доступа URL: https://developer.android.com

21. Официальный сайт компании Jetbrains [Электронный ресурс]: Среда разработки. - Режим доступа URL: http://jetbrains.ru/

22. Официальный сайт компании ООО «Орион Телеком». [Электронный ресурс]: информация об истории компании. - Режим доступа: URL:http://orionnet.ru/krk

23. Пауэрс, Ш. Изучаем Node.js: пер с англ./ Ш. Пауэрс. - Санкт-Петербург: Питер, 2014. – 400 с.

24. Прамодкумар, Д. NoSQL: новая методология разработки нереляционных баз данных.: пер с англ. / Д. Прамодкумар, Садаладж, М. Фаулер; под ред. С.Н. Тригуб. – Москва.: Вильямс, 2013 - 192 с.

25. Свободная энциклопедия [Электронный ресурс]: SHA-2. / «Википедия». - Режим доступа: https://ru.wikipedia.org/wiki/SHA-2

26. Советов, Б.Я. Информационные технологии.: учебное пособие / Б. Я. Советов, В. В. Цехановский. – Москва.: Высш. шк., 2003. - 263 с.

27. Справочные материалы по информационным технологиям. [Электронный ресурс]: Методология IDEF0. / «АйтиТич». - Режим доступа: URL:http://itteach.ru/bpwin/metodologiya-idef0/all-pages

28. Сухов, К. HTML5 – путеводитель по технологии.: учебное пособие / К. Сухов. – Москва.: ДМК Пресс, 2013. – 352 с.

29. Флэнаган, Д. JavaScript. Подробное руководство 5-е изд.: пер с англ. / Д. Флэнаган; под ред. А.Галунов – Санкт-Петербург.: Символ плюс, 2008. – 1080 с.

30. Флэнаган, Д. JavaScrrt: карманный справочник, 3-е изд.: пер с англ. / Д.Флэнаган; под ред. С.Н. Тригуб – Москва.: Вильямс, 2013. – 230 с.

31. Хавербек, М. Выразительный JavaScript. 2-е изд.: пер с англ. / М.Хавербек; под ред. В.Голованова. – Москва – Вильямс, 2014 – 437 с.

32. Хорев, П.Э. Объектно-ориентированное программирование с примерами на C# .: учебное пособие / П.Б. Хорев. – Москва.: Инфра-М, 2016. – 200 с.

33. Шаши, Ш. Основы построения баз данных.: пер с англ. / Ш. Шаши. – Москва.: КУДИЦ-ОБРАЗ, 2004. – 336с.

34. Шмидт, К. HTML5. Рецепты программирования.: пер с англ. / К.Шмидт, К.Симпсон; под ред. А.Кривцов. – Санкт-Петербург.: Питер, 2012. – 288 с.